13장 스코프

13.1 스코프란?

자신이 선언된 **위치**에 의해 다른 코드가 식별자 자신을 참조할 수 있는 **유효 범위**

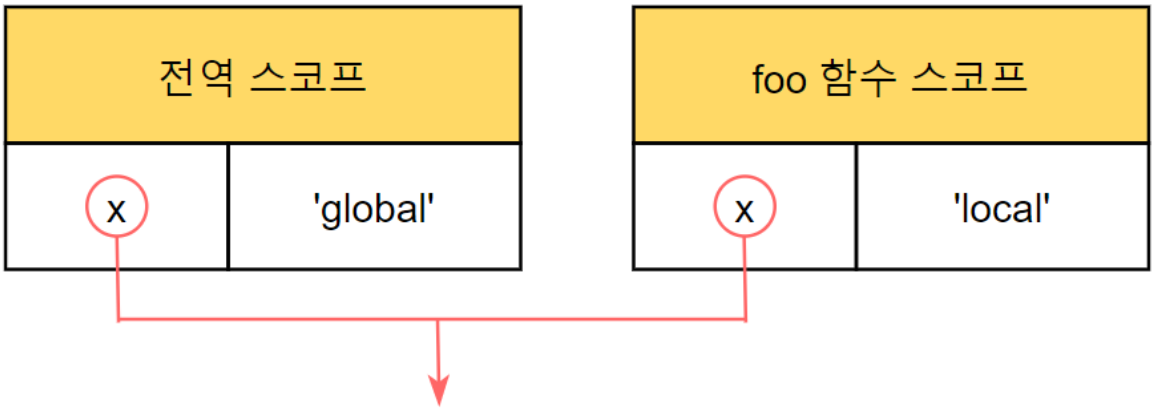
⇒ 식별자가 유효한 범위

```
var x = 'global';

fnuction foo() {
    var x = 'local';
    console.log(x); // (1)
}

foo();

console.log(x); // (2)
```



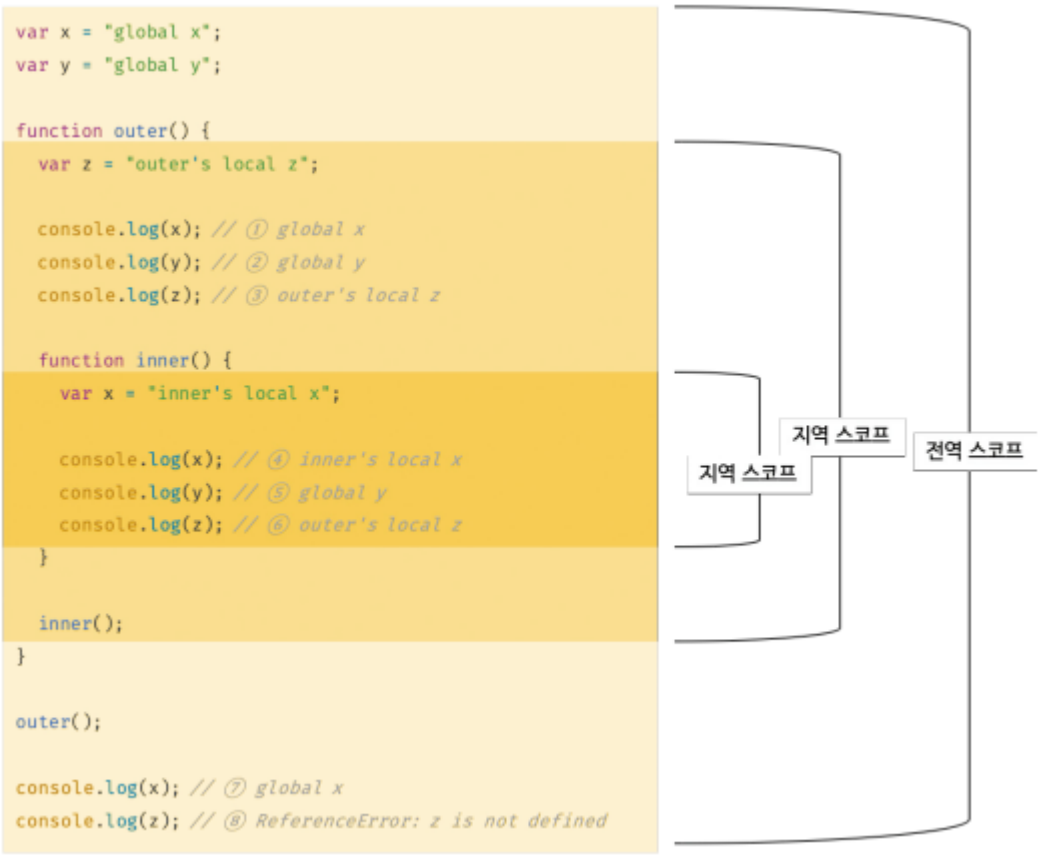
https://velog.io/@hang_kem_0531/TIL-모던-자바스크립트-Deep-Dive-스코프

- 같은 이름을 갖는 변수는 충돌을 일으킨다.
- 그러나 하나의 파일 이름만 사용해야 한다면 번거롭다.
- 따라서 스코프(유효 범위)를 통해 같은 이름의 변수를 사용할 수 있게 한다.

💡 var 키워드로 선언된 변수는 중복 선언이 허용되며, let과 const는 허용하지 않는다.

13.2 스코프의 종류

구분	설명	스코프	변수
전역	코드의 가장 바깥 영역	전역 스코프	전역 변수
지역	함수 몸체 내부	지역 스코프	지역 변수



https://velog.io/@jinseoit/js-scope

- 전역 변수 : **어디서든지** 참조할 수 있다.
- 지역 변수 : 자신의 지역 스코프와 하위 지역 스코프에서 유효하다.

13.3 스코프 체인

- 함수는 중첩될 수 있으며, 따라서 함수의 지역 스코프도 중첩될 수 있다.
    - => 스코프가 함수의 중첩에 의해 계층적 구조를 갖는다.

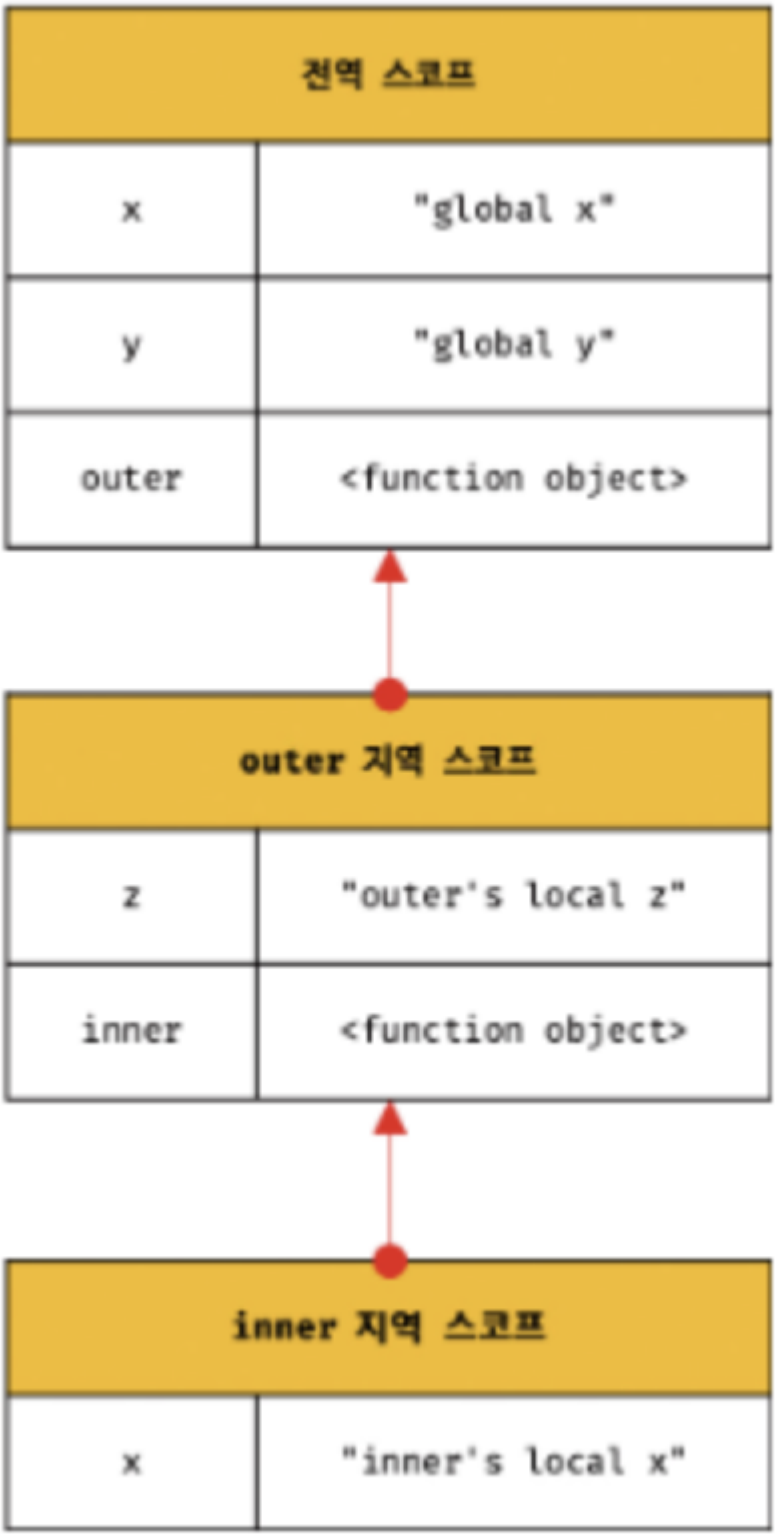


그림 13-3 스코프 체인

https://velog.io/@myday0827/모던-자바스크립트-13장

- **스코프 체인**

- 스코프가 계층적으로 연결된 것
- 스코프 체인을 통해 변수를 참조하는 코드의 스코프에서 **시작**하여 **상위 스코프 방향**으로 이동하며 선언된 변수를 검색

> 💡 **렉시컬 환경**
> 스코프 체인의 물리적인 실체
>
> **전역** 렉시컬 환경은 코드가 로드되면 곧바로 생성되며,
>
> **함수** 렉시컬 환경은 함수가 호출되면 곧바로 생성된다.

13.4 함수 레벨 스코프

- 지역은 함수 몸체 내부를 말하기 때문에 **지역 스코프**는 **함수**에 의해서만 생성된다.
- 대부분의 프로그래밍 언어 및 let, const 키워드는 함수 몸체만이 아닌 모든 코드 블록에서 지역 스코프를 만들고, 이를 **블록 레벨 스코프**라 한다.
- 그러나 **var 키워드**로 선언된 변수는 오로지 **함수의 코드 블록(함수 몸체)**만을 지역 스코프로 인정하며, 이를 **함수 레벨 스코프**라 한다.

```
var x = 1;

if(true) {
    // 함수 밖에서 var 키워드로 선언된 변수는 코드 블록 내에서 선언되었다
    // 따라서 x는 전역 변수다. 이미 선언된 전역 변수 x가 있으므로 x 변수
    // 이는 의도치 않게 변수 값이 변경되는 부작용을 발생시킨다.
    var x = 10;
}

console.log(x); // 10
```

13.5 렉시컬 스코프✨

```
var x = 1;

function foo() {
    var x = 10;
    bar();
}

function bar() {
    console.log(x);
}

foo(); // ?
var(); // ?
```

- 동적 스코프 : 함수를 어디서 **호출**했는지에 따라 함수의 상위 스코프를 결정한다.
- 렉시컬(정적) 스코프 : 함수를 어디서 **정의**했는지에 따라 “
  - 자바스크립트는 렉시컬 스코프를 따른다.
  - 이 후 함수가 실행될 때 결정된 상위 스코프를 기억한다.